Fast Time Plot Data Acquisition

Faster than 15 Hz

Fri, Jan 14, 1994

The analog IndustryPack module used in the Internet Rack Monitor (IRM) includes support for recording 1 KHz samples of 64 A/D channels in a 64K byte circular buffer. The buffer wraps in 512 ms. It is desired to access such data for the purpose of making plots. In the Acnet control system, the Fast Time Plot protocol (FTPMAN) is used to acquire this data from the front ends. A console requests data to be delivered at 15 Hz down to 2 Hz. In principle, the console could collect 1 KHz data, but it limits its Continuous mode plotting support to 720 Hz data. In the "Auto" mode, the limit is 200 Hz. Faster rates must be accessed via the Snapshot mode.

The SSDN that is sent in FTP's Timing Info Request message is the same SSDN that is used when the named device is called up for display on a Parameter Page. In order to access the data from the analog IP board, FTPMAN must somehow derive the class code that describes what plotting support is allowed for the given signal. To deliver the plotting data, both the 64K memory that holds the 1 KHz data as well as the register block for the IP board must also be determined. But the SSDN now contains only the analog channel number that points to the data pool.

One possibility would be to utilize a spare byte that has so far not been used in the SSDN data structure. This would require changing the off-line uploading program that updates the Acnet database with changes made in a local station. How can the uploading program determine in what set of 64 channels a given channel resides, if indeed it resides in *any* set of 64 channels from an analog IP board. One could use a spare byte in the analog descriptor for this purpose. And one could assume that the low 6 bits of the channel number indicate which channel out of a 64-channel set is being referenced. It would seem wasteful to use the spare byte for this purpose, however, as each channel in a 64-channel range would have to be marked in the same way. Perhaps a small table of channel # ranges could relate to the memory and register block locations.

A simpler approach might be to fix the ranges of channels used for available IP boards. An IRM can have one or two such boards. But a VME-based station that uses IP carrier boards might have more.

Another concern is to provide support for up to 1 KHz data via a new listype. In this case, an ident format must be defined. It could include the type of 64-channel block. But how can the user enter this information, say, on the Macintosh Parameter Page? The support code for this listype could

returned for each ident. The internal pointer could include the 16-bit offset of the next datum to be collected, so that each update would begin where the last one ended in access to the circular memory buffer.

A simple approach can be as follows: Assume that channel#s `0100–013F` are supported by the IP module in slot *d*, in which the base address of the memory is `00630000` and the register block is at `FFF58300`. For channel#s in the range `0140–017F`, assume the IP module is in slot *c*, the memory base address is `00620000` and the register block location is `FFF58200`. For VME-based stations that do not use IP modules, these channels could not be assigned; at least no one could try to FTP them. This scheme could work for IRMs that have 64 or 128 channels.

In order to collect data that is measured at times relative to clock events, a different plan should be adopted. If the reply data includes a 32-bit time associated with the first point, and a second 32-bit number that is the time between the two most recent events used for this request, then including a timestamp with each data point that is the time measured from the most recent one of those events would allow for 16-bit timestamps. The host program would need to add each time offset to the time of the first point to get the plotting time value. If this sum is greater than the given event time difference, then subtract this event time difference from the sum to get the time value suitable for plotting the data point. Note that by definition the first time offset would be zero.

Assume that we get an interrupt whenever a clock event is written into a FIFO. In response to the interrupt, sample the 1 MHz free-running timer on the MCchip on the 162 board and save it in an array of 256 entries, one for each clock event. Besides the sampled time of each event, also measure the time between such events. This time between the two most recent events would be used in replies for the plot data described above.